A Hierarchical Orderly Layout Algorithm Based on Channel Points

Yikun Zhang[1], Kaibao Hu[1], Xinhong Hei[1], Ming Zhao[2], Minghui Zhang[1] and Hao Chen[3]

[1]School of Computer Science & Engineering, Xi'an University of Technology, Xi'an, 710048, China
[2]Shanghai Zero One Assembly Software Company Limited, Shanghai, 200120
[3]School of Mechanical Instrumental & Engineering, Xi'an University of Technology, Xi'an, 710048, China
ykzhang163@163.com

Abstract - Since the conventional hierarchical layout algorithm has the disadvantage of wiring chaos in large programs, so on the basis of Sugiyama layout algorithm, this paper proposes a hierarchical orderly layout algorithm based on channel points. Adjust the number of each node in the channel dynamically according to program size in order to solve the existing algorithm's overlap problems when wiring lines, Combine the generalized tensor balance idea in layout diagram to reduce cross and realize the artistic layout, the algorithm also give the corresponding line distribution and application strategy in accordance with the relative positional relationship between the calling nodes to achieve the routing orderly. Experiment results show that the algorithm has good layout efficiency. It can reduce the crossings effectively, be realized layout clearly and be implemented simply.

Index Terms - Software visualization, hierarchical graph, generalized tensor balance algorithm, crossing minimization, channel routing.

I. Introduction

With the development of the software system, the development, testing and maintenance of the software system is becoming more and more difficult. Software quality assurance has become one of the issues of growing concern [1-2]. When the visualization technology is applied into the program, the software product's understandability, testability and maintainability can be greatly improved.

In module call graph, since the function call and the class inheritance have obvious features, and hierarchical graph performance is relatively simple and intuitive, the diagram uses hierarchical layout. However, the layout problem itself is a NP problem [3]. The layout constraints can easily lead to conflict [4]. Such as the figure's minimum height and aspect ratio of coordination, the length of the side optimization, minimize the number of edge crossings, etc. Therefore it is necessary for us to weigh the various constraints, and ensure that the hierarchical graph correspond to the traditional aesthetic standards.

The current layout algorithms mention some algorithms [5-7],which tell us how to reduce the cross and how to connect nodes. But it considers little about the size of the program (node number, such as the number of the function in the program or the number of class), which does not have good applicability. And for large-scale software systems, the call relationship complex node's connection is confusing even overlapping, and the node with smaller call relationship has a lot of space, which brings much inconvenience for the visualization and software development and maintenance. In this paper, we combine several algorithms, and propose a hierarchical orderly layout algorithm based on channel points. The readability and understandability of the hierarchical graph are enhanced.

II. Hierarchy Layout Algorithm Flow Chart

Layout consists of two parts, the layout and the wiring. In the graphical layout, the most widely used method is the Sugiyama layout algorithm [8]. It consists of three modules, level specified, cross reduction and coordinates specified. This paper starts from the Sugiyama layout algorithm. In order to solve the large-scale program wiring confusion overlap problem when conducting the layout operation, the node's channel point layout and the channel point line are increased. The number and distance of the channel point are dynamically adjusted according to the program size. Besides, the relationship node routing strategy based on the channel point is given. Fig.1 shows the basic algorithm process.

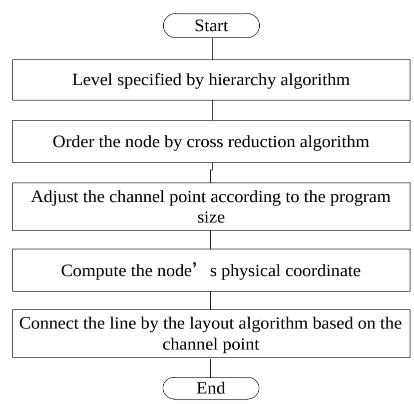

Fig. 1 The Hierarchy layout algorithm flow based on Sugiyama

III. Hierarchy layout algorithm

In this paper, the layout of the hierarchy diagram is divided into four steps. They are node level specification, node sorting between the layers, node's channel point specification and node's physical coordinate calculation.

A. Node level specification

When conduct the layout operation on the hierarchy graph, firstly, we need to specify the node's level. Namely, confirm its layer according to the call relation between the nodes. Connection type is from top to bottom while wiring. So, the layout is from top to bottom. Take the call node to the upper, and the called node is placed on the lower levels. The current hierarchical algorithm mainly includes the longest path hierarchical algorithm, the minimize width hierarchical algorithm, and the virtual node minimization algorithm [9].

B. Node sorting between the layers

Reduce cross is considered one of the most basic problems in the hierarchical graph layout. Cross is caused by the relative sequence of the calling node position. The reduction of the cross point makes it more clear for the user. So, after conducting the layout operation, it is necessary for us to reordering the node for each layer to reduce the cross [10]. In the cross reduce algorithm, generally, we use 2- layer cross reduction algorithm to reduce the cross number of the input layer [9]. Consider from the aesthetics and readability of the layout aspect, the generalized tensor balanced thought is used in this paper [11]. Take the node with large correlation to the center position of the layer, and the node with small correlation is taken to the side to achieve the symmetrical balance of the graph.

The calculation formula of *column_(id)* is shown as follows:

$$column\_(id) = (layer\_width + 1) / 2 + d * s \tag{1}$$

Layer_width is the maximum width of the layer, (layer_width + 1) / 2 is the center position of the layer, d is the center point offset of the next node. Iinitially, d = 0, s is relative to the location of the center node (the left is -1, the right is 1), s = 1 initially.

Calculate the first layer node with formula (1) directly. For each calculation d+1, and s is 1 and -1 in turn, If it is not the first layer node, we also need to consider the node's call set and the layer difference and the column difference, s is 1 and -1 respectively. Calculate the node's total difference *total_dis,* and take the node to the column with the minimum value of the total difference to reduce the number of the cross point and the length of the connection line.

The calculation formula of the total difference *total_disis* expressed as follows:

$$total\_dis = |\,(layer(i) - layer(j)\,) * (column(i) - column(j)\,)\,| \tag{2}$$

Here, layer (i) - layer (j) is the layer difference of the node i and node j, column (i)-column (j) is the column difference of the node i and node j.

C. Node's channel point specification

Calculate the node's absolute physical coordinates by the position of the starting point, node layer and node column. The existing layout algorithm considers little about the program size. It adopts the virtual node to connect when cross layer connecting. This approach greatly increases the sort time of the node (when sorting node contains a virtual node). Besides, the wiring space between the nodes is limited. Then the layout loses balance, and the aesthetics and readability are affected.

In this paper, we increase the algorithm flow of channel point specification, introduce the program size (node_size), channel point number (bit_num) and the distance between the channel point (line_dis). Formula (3) shows us the detailed method to adjust the number and the distance of the channel point according to the program size. Meanwhile, in order to reduce the sorting time and enhance adaptability of the algorithm, we use the letter i (idle) and u (occupied) to mark the status of each channel point.

$$\begin{cases} bit\_num = 5, line\_dis = 3 & (node\_size < 120), \\ bit\_num = 8, line\_dis = 2 & (120 <= node\_size <= 500), \\ bit\_num = 15, line\_dis = 1 & (node\_size > 500), \end{cases} \tag{3}$$

The channel point distribution diagram of node i is shown in Fig.2. According to the program size, in the limited space, the number and distance of the channel point is dynamically adjusted. We use i and u to mark the routing condition. If the routing is idle, then it will be marked i, otherwise u.

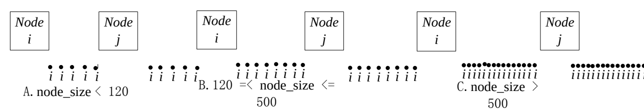

Fig. 2 Channel point distribution diagram of node i

D. Node's physical coordinate calculation

Section *A*, *B* and *C* discussed the node layout, and the relative layout of the node is obtained. The next step is to take the node location into the specified display form. Namely, calculate the location coordinate of the node. Choose the rectangular frame shown as a graph element. According to the position of the starting point (TOP LEFT), the layer of the node i layer (i) and the column (i) value, get the position coordinate (top,left) of the node i by formula (4).

$$\begin{cases} top = TOP + layer(i) * (BOX\_H + BIT\_H), \\ left = LEFT + column(i) * (BOX\_W + BIT\_W), \end{cases} \tag{4}$$

Here, BOX_H means the height of the rectangle, BIT_H is the height of the channel point, BOX_W stands the width of the rectangle, and BIT_W represents the width if the channel point.

IV. Channel Routing Strategy

After the node layout, the next step is to wiring the node. Wiring means generates the layout diagram based on the call relationship between the nodes. In accordance with this article, according to the node sorting thought of generalized tensor balance, we adopt the wiring way. One is from the primary to the secondary, and the other is from the center to both sides. Firstly, we can consider the wiring of the main module, namely, the nodes characters large correlation. By this way, the node possesses the optimal channel. Then, consider the secondary module wiring. When a node channel line is fully occupied, use different wiring strategies according to the node's relative position.

According to the difference of node i and node j layer_dis (i, j), the wiring strategy is divided into the following three:

$$\begin{cases} (a)\ layer\_dis(i,j)=0,(\text{the same layer connection}) \\ (b)\ |layer\_dis(i,j)|=1,(the\ \ adjacent\ \ layer\ \ connection) \\ (c)|layer\_dis(i,j)|>1,(the\ \ cross\ \ layer\ \ connection) \end{cases}$$

(a) The same layer connection.The same layer connection node is also called self-call node. We can draw an arc on this node pixel rectangle to mark the node.

(b) The adjacent layer connection。For the adjacent layer connection, we can directly use the segment to connect regardless of the connection through the node. In this paper, we adopts the solid line and the dotted line to represent the call relationship between the node, remove the traditional algorithm, in which case the arrow is used to stands the call relationship between the nodes (the solid line represents the top-to-bottom call, the dotted line represents a call from bottom to top). In a large-scale program, the call relationship is very complex, it seems confusing when wiring use the arrow. It is difficult for us to know which node is the invoke node and which node is called node.

(c) The cross layer connection. According to the column number column_(i) and column_(j) of the call node i and node j, the wiring is divided into two types, one is cross layer in the same column, and the other is the cross in the different column. And, for the cross layer in the same column, it contains three conditions. In this paper, we adopt the dynamically allocation and application for cross layer wiring. Obtain the connection line from the node routing or the adjacent node channel line. Besides, the routing is marked as u (occupied) to achieve the wiring orderly.

(1) Cross-layer in the same column: column_(i) = column_ (j). If column_(i)<=layer_center, apply an idle routing from the channel point of the column_(i)-1 column. If there does not have any idle line, apply from the right side column of the column_(i)-1. If column_(i)>layer_center, apply an idle routing from the channel point of the column_(i) column. If there does not have any idle line, apply from the left side column of the column_(i). Fig. 3 is the wiring diagram of across layer in the same column.5-(1) represents the left side of the display form center, and the 5-(2) is the right side.

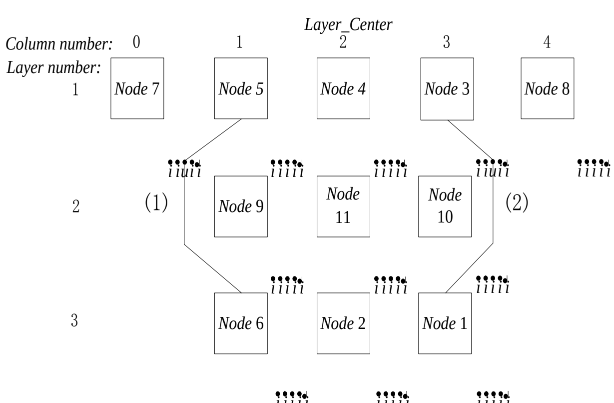

Fig. 3 The wiring diagram of across layer in the same column

(2) Case 1 of cross layer in different column: if (column_(i) < column_(j) && column_(j) <= layer_center), then apply the idle line from the channel point of column_(i). If there does not have any idle line, apply from the right side column of the column_(i) (shown as Fig.4).So, if (column_(j) >= layer_center && column_(i) > column_(j)), then apply the idle line from the channel point of column_(i)-1. If there does not have any idle line, apply from the left side column of the column_(i)-1(shown as Fig.4).

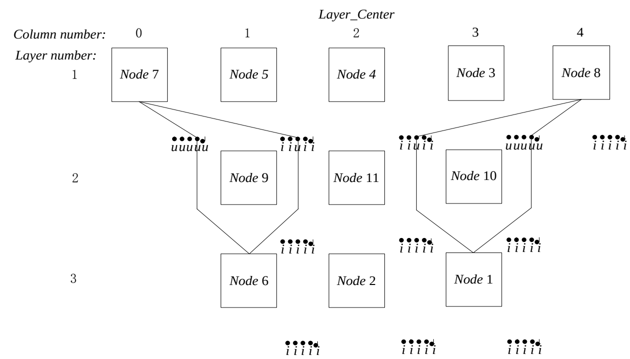

Fig. 4 The wiring diagram of across layer in different column of case 1

(3) Case 2 of cross layer in different column: if column_(i) < column_(j), then apply the idle line from the channel point of column_(j)-1. If there does not have any idle line, apply from the left side column of the column_(j)-1 (shown as Fig.5). If column_(i) > column_(j)), then apply the idle line from the channel point of column_(j). If there does not have any idle line, apply from the right side column of the column_(j) (shown as dotted line of Fig.5).

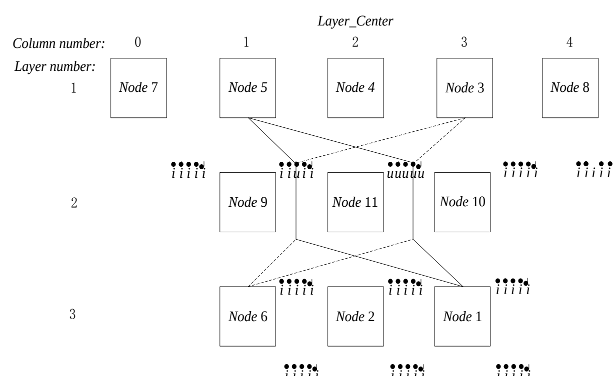

Fig. 5 The wiring diagram of across layer in different column of case 2

(3) Case 3 of cross layer in different column: if (column_(i) >column_(j) && column_(i) <= layer_center_, then apply the idle line from the channel point of column_(j). If there does not have any idle line, apply from the right side column of the column_(j)(shown as Fig.6). If (column_(i) <column_(j) && column_(i) >= layer_center_, then apply the idle line from the channel point of column_(j)-1. If there does not have any idle line, apply from the left side column of the column_(j)-1 (shown as dotted line of Fig.6).

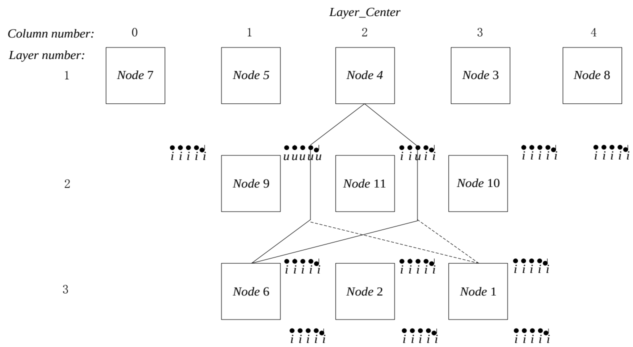

Fig. 6 The wiring diagram of across layer in different column of case 3

The channel routing algorithm realizes the orderly distribution lines, and implements the layout of the uniform and local symmetry.

V. Experimental Results

Take Fig.7 as example, we suppose that the program size is less than 120, and conduct the cabling operation on node i and node j by using the traditional layout algorithm and the proposed algorithm respectively. The layout diagram is shown as Fig. 7-a and Fig. 7-b. Here, node i has 13 call relationships. 3 is the adjacent layer calls, and the remaining 10 is the cross layer calls. For node j, it has 4 call relationships. 1 is the neighbor layer call, and the remaining 3 is the cross layer calls.

For Fig.7-a, the cross layer connection is realized by increasing the virtual node to connect. When across a layer, a virtual node is added, thus the node ordering time is increased, which leads to the inefficiency in wiring. And, the algorithm does not take into account the size of the program and the size of the display area. Besides, it also does not allocate and adjust the routing effectively. So, when the call relationship of node i is complex, the wring loses balance.

For Fig.7-b, the channel point number for each node is dynamically assigned to 5 according to the program size. It is not necessary to consider the channel node number when ordering the nodes. Use the different wiring routing strategy according to the call node relationship, to make the layout even symmetrical. Below the channel line, we use i (idle) and u (occupied) to mark the routing, and adopt the numeral to stand the requested order of the routing, and then reach the orderly wiring. The routing algorithm has good applicability.

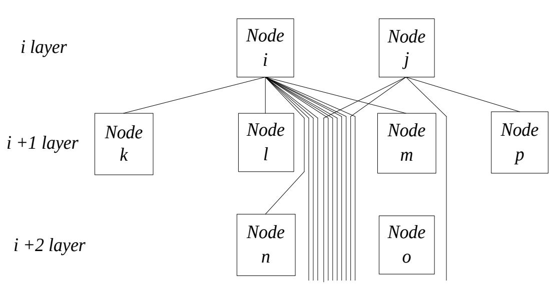

Fig. 7-a

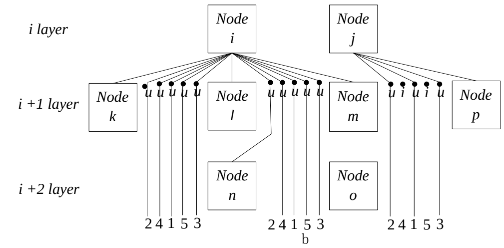

Fig. 7-b

Fig. 7 The contrast diagram of the two algorithms

VI . Conclusions

The hierarchical layout algorithm proposed in this paper implements the visualization of software module call diagram (function call relations and class inheritance relations). Since the node call relationship is different, this paper adopts the corresponding wiring strategy to allocate and apply the connection lines, achieve the wiring orderly. The layout algorithm features uniform wiring, symmetry, etc., which enhances the readability of the visualization diagram.

Acknowledgment

The authors would like to thank the anonymous reviewers. This work is supported in part by the National Natural Science Foundation of China (Grant No. 61100173), and Shaanxi Provincial Higher Education Reform key project (Grant No. 11J15).

References

[1] Angus G. Yu. Software Crisis, What Software Crisis?. 2009 International Conference on Management and Service Science. Washington, DC:IEEE Computer Society, 2009:1-4.

[2] Vladimir Getov. Software Development Productivity:Challenges and Future Trends. 34th Annual IEEE Computer Software and Applications Conference. Washington, DC: IEEE Computer Society, 2010:2-7.

[3] MARTIR. Arc crossing minimization in graphs with GRASP. IIE Transactions, 2001, 33(10):913-919.

[4] Wang XB, Wang H, Liu C. Automatic hierarchical layout algorithm for UML class diagram. Journal of Software, 2009, 20(6):1487-1498.

[5] ZHANG Yikun, ZHU Wei, WANG Kai, HU Yanjing. Hierarchical layout algorithm based on order of succession and degree Of association. Journal of Computer Applications, 2009, 29(5):1373-1375.

[6] Michael Ogawa, Kwan-Liu Ma. Code_swarm: A Design Study in Organic Software Visualization. IEEE Transactions on Visualization and Computer Graphics, 2009, 15(6):1097-1104.

[7] Martin Harrigan, Patrick Healy. Efficiently drawing a significant spanning tree of a directed graph, 6th International Asia-Pacific Symposium on Visualization. Washington, DC: IEEE Computer Society, 2007: 53-59.

[8] Christian Bachmaier. A Radial Adaptation of the Sugiyama Framework for Visualizing Hierarchical Information. IEEE Transactions on Visualization and Computer Graphics, 2007, 13(3):583-594.

[9] SUN Junhuan. Research on man-machine interface in technology dedicated system . Heilongjiang:Harbin Engineering University, 2009:22, 25.

[10] ZHAO Guoqing, YANG Nanying, JIA Zhenyang, FAN Dian, HUANG Ronghuai. Research on Algorithm for Drawing Concept Maps . Journal of Open Education Research, 2005, 11（5）:32-37.

[11] SUN Chang-ai, LIU Chao, JIN Mao-zhong. Effective Wove Algorithm for Software Structure Graph. Journal of Beijing University of Aeronautics and Astronautics, 2000, 26(6):706-708.